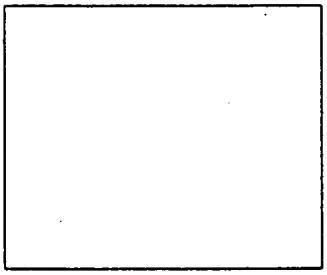
Appendix B

Title: Universal Serial Bus Device Controller 1.1

3 Overview



The Universal Serial Bus Device Controller (USBDC) interfaces the USB Function Device to the Universal Serial Bus. The USBDC handles the USB protocol and provides a Flexible Peripheral Interconnect Bus (FPI) interface on the local side. The internal Standard Microcontroller Interface (SMIF) provides a simple Read/Write protocol that allows an easy adaptation to other local bus protocols.

The USBDC Bus runs on a 12 MHz Clock which is extracted from an external 48 MHz clock on USB side. On the FPI side the USBDC is designed for up to 66 MHz. An internal FIFO is used for clock and data rate adaptation.

The USBDC can be configured in terms of number of configurations, interfaces and EndPoints depending on the device' s requirements.

3.1 Features

Some of the features of the USBDC are:

- Compliant with USB protocol revision 1.1.
- Support for both low speed and full speed devices.
- USB protocol handling.
- No microcontroller or firmware involved.
- USB device state handling.
- Clock and data recovery from USB.
- Bit stripping and bit stuffing functions.
- CRC5 checking, CRC16 generation and checking.
- Serial to parallel data conversion.
- Maintenance of data synchronization bits (DATA0/DATA1 toggle bits).

Type	Package
PEB XXXX	

- Supports up to three configurations with each configuration supporting four interfaces and each interface handling up to four alternate settings (Option to go up to eight alternate settings is also provided under certain circumstances).
- Programmable number of physical endpoints and supports up to 16 bi-directional logical endpoints.
- User configurable endpoint information.
- Understanding and decoding of standard USB commands to endpoint zero.
- Option to decode the GetDescriptor command or to pass the command onto the application bus.
- Support of Class/Vendor commands by passing the Setup transactions to the application bus.
- Support of string descriptors.
- USB clock: 12 MHz
- System clock: up to 70 MHz
- Number of gates:
- Area:
- Power consumption:
- Full scan path and BIST of on-chip RAMs for production test

3.2 Logic Symbol

3.3 Typical Applications

5 Functional Description

5.1 Functional Overview

The USB1 provides the USB functionality for the TriCore platform. This module provides a 12Mb/s full speed USB Vers. 1.1 compliant interface. It interfaces the USB transceivers on the one end and the Flexible Peripheral Interconnect bus (FPI) on the other end. Implemented as a slave device, it communicates with the Peripheral Control Processor (PCP) via side band signalling using interrupt signals.

The module provides the following functionality:

- USB device interface, compliant to USB specificatin, version 1.1.
- 12Mb/s full speed interface
- Support for audio, data and communication device classes.
- Two different configurations besides default configuration 0.
- One Bi-Directional control endpoint 0 and 15 SW-configurable unidirectional endpoints.
- Isochronous packet size up to 1023 bytes, control, bulk and interrupt packet size up to 64 bytes.

5.2 Block Diagram

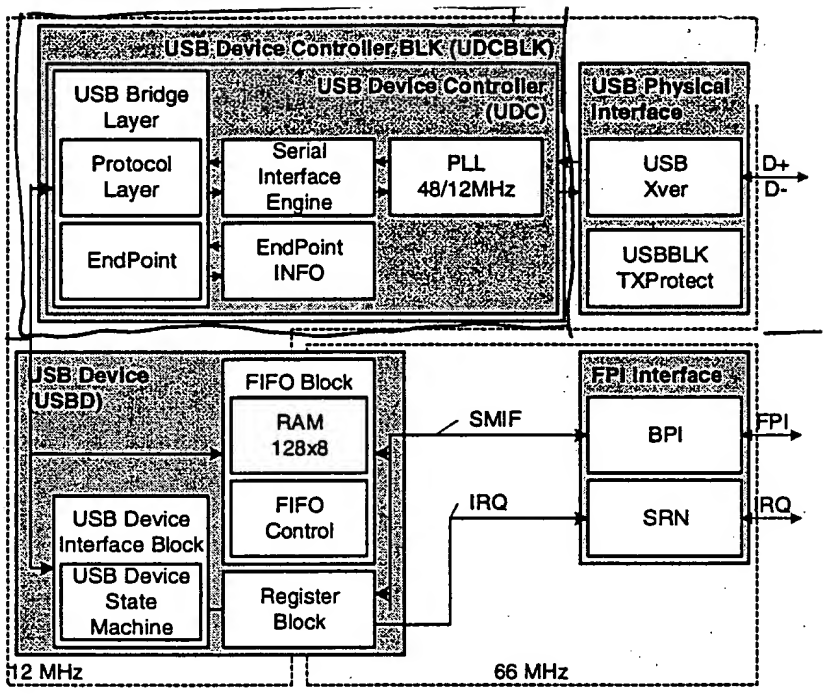


Figure 1 USB1 Block Diagram

5.3 Functional Blocks

The following paragraph gives an overview of the functions implemented in the macro's blocks.

For each block an indication about the changes to the existing UTAH's USBBLK is given.

5.3.1 USB Device Controller (UDC)

The USB Device Controller is a macro provided by Phoenix Technologies Ltd. Except of the configuration settings the Macro itself is not changed in this application. The following paragraph gives a overview of the UDC as it is described in the USB Device Controller User's Manual Rev. 2.0 [7] for detailed information see the referenced document.

The Universal Serial Bus Device Controller Core (UDC) interfaces the USB Function Device to the Universal Serial Bus. The UDC handles all the USB protocol and provides

a simple Read/Write protocol on the Function Interface (Application Bus). The UDC and the Application Bus run on a 12 MHz Clock which is extracted from the 48 MHz clock provided by the User (FullSpeed Applications). The UDC does not have any FIFOs built in which helps the user to define his/her own FIFO requirements. The UDC is configurable in terms of number of configurations, interfaces and EndPoints depending on the Device' s requirements.

This block and its subblocks needs to be recompiled for the desired amount of Interfaces/endpoints. The necessay patch should be available at Infineon. All figures concerning number of interfaces and endpoints need to be validated.

The main blocks in the USB Device controller are:

5.3.1.1 PLL Block

The PLL Block has a digital PLL built-in. The main functionality of the PLL Block is to extract the USB Clock and Data from the USB Cable. The input to the PLL Block comes from an external differential transceiver. The PLL runs on a 48 MHz clock for a full speed configuration. The PLL Block also generates a 12 MHz clock from the 48 MHz clock and can optionally supply to the SIE and UBL Blocks. The PLL identifies the single ended zero (SE0) signal on the USB and sends it to the SIE block. In the low speed configuration the PLL runs on a 6 MHz clock and generates a 1.5 MHz clock.

No modification necessary

5.3.1.2 Serial Interface Engine Block (SIE)

The second block is the Serial Interface Engine block (SIE) that does all the front end functions of the USB protocol such as SyncField identification, NRZI-NRZ conversion, token packet decoding, bit stripping, bit stuffing, NRZ-NRZI conversion, CRC5 checking and CRC16 generation and checking. The SIE also converts the serial packet to 8 bit parallel data. The SIE block has a one byte buffer built-in for buffering the data during data transmission and reception.

Recompilation may be necessary

5.3.1.3 USB Bridge Layer (UBL)

The third block is called the USB Bridge Layer (UBL) block that handles the error recovery mechanism during transactions while interfacing to the application. The UBL also decodes and handles all the standard control transfers addressed to EndPoint zero. The UBL passes all the Vendor/Class commands onto the Application Bus so that the application can decode the command and act there upon. This provides the flexibility of using the UDC core in multiple applications. The UBL supports up to a maximum of 3 configurations with each configuration having a maximum of 4 interfaces. Each interface can have up to 4(8) alternate settings. The UBL block has two sub blocks called the Protocol Layer (PL) and the EndPoint (EP) block.

Recompilation necessary

Protocol Layer (PL)

The PL Block controls the SIE Block by providing necessary Handshake signals to the SIE and talks to the Application Bus Logic. It also has the mechanism for error recovery if the Application Bus violates the data transfer protocol. The Application Bus protocol is explained later in the manual.

End Point (EP)

The EP block handles all the control transfers to EndPoint zero. The EP block decodes and responds to all the USB standard commands and passes the USB Class/Vendor commands to the Application Bus. The EP block maintains the buffer for device address, buffer for storing the present active configuration, and the logic to determine the present state of the device.

5.3.1.4 EndPoint Information Block (EPINFO)

This is a configurable block in the UDC Core that can be configured by the user by using the UDC Compiler (RapidScript). This enables the UDC to be configured for different applications and tailors to the needs and requirements of different applications. The EPINFO block maintains the buffer (registers) that stores the information about all the EndPoints the device supports. This also stores the information about the size of configuration and all string descriptors that are supported in the UDC core. The information about the current EndPoint is multiplexed from these buffers and is provided to the PL block which inturn will control the SIE block based on this information. The EPINFO also has the DATA0/DATA1 synchronization bits for each bidirectional EndPoint the UDC supports. The EPINFO also has the EndPtStalled Bit for each of the supported logical EndPoint to indicate the stalled status of the EndPoint. The EPINFO supports up to 16 active bidirectional Logical EndPoints. The EndPoint Number ranges from 0 to 15. The number of Physical EndPoints is programmable.

Recompilation necessary

5.3.2 USB Device Controller Block (UDCBLK)

The UDCBLK is a wrapper around the UDC, where the following signals are connected:

- pllreset and reset: Connected to x_res_n even if the UDC specification says that the pllreset must be taken away at least two clock cycles before the reset is taken away. This is possible because the synchronous reset into the UDC was modified into an asynchronous reset of all the registers.
- dev_clk_gated is simply connected to udc_clk - in the UDC manual they say there is a possibility to gate this clock signal with gate_dev_clk but we don't use this possibility

because the chip itself has the capability of turning off the clock for the entire UDC in suspend mode.

Changes that were made to the UDC:

- All resets of registers which can be reset changed from synchronous to asynchronous reset.
- Added a asynchronous reset to all registers except in FIFO and DPLL modules.
- In scan mode switching of the clocks in the module DPLL.
- In scan mode turning off the asynchronous set of a register in the SUSP_RES module.
- Added an input DEV_Milisec for generating an millisecond pulse in the SUSP_RES module by writing into a dedicated register in order to reduce the simulation time of a suspend/resume test. A write to this register fakes the assignment of a millisecond pulse which is usually generated by a counter.

Slight modifications due to extended pinout of the UDC

5.3.3 USB Physical Interface (UPHY)

This block contains the circuitry necessary to fulllfill the electrical specification of the USB. It contains the USB Transceiver (USBXVER) and the USB transceiver protection (USBBLK TXPROTECT). For details please see the USB specificatin, Chapter 8 [1)]

No modifications necessary

5.3.3.1 USB Transceiver (USBXVER)

This block contains the differential transmitter that is used for driving the USB data signal onto the USB cable and the differential receiver used for receiving the data from the cable.

Migration from C7 (UTAH) to C9 (Harrier-VT)

5.3.3.2 USBBLK TXPROTECT

This block is for protecting the USB transceiver from shortcuts on DPLS and DMNS. Th USB specification[1)] requires, that the transceiver which is connected to DPLS and DMNS (which is at 5V) must resist a short-cut of DPLS and DMNS to VDD or GND for a long time. In order to prevent the transceiver from being damaged, it will be turned off while a shortcut is detected.

No modifications necessary

5.3.4 Flexible Peripheral Interconnect Interface (FPIIF)

The Flexible Peripheral Interconnect Interface (FPIIF) contains a generic Bus Peripheral Interface (BPI) that interconnects the Flexible Peripheral Interconnect bus (FPI) and the Slave Module Interface (SMIF). It also contains a Service Request Node (SRN) for interrupt arbitration to the external PCP.

5.3.4.1 Bus Peripheral Interface (BPI)

Please see the Design Documentation BPI for FPI, Version 3.10 [10] for a brief description of the module's functionality.

Generic Macro has to adopted.

5.3.4.2 Service Request Node (SRN)

Please see the TriCore Architecture Manual [12] and the Microcontrollers ApNote AP3222 [13] for a brief description of the module's functionality

Generic Macro has to be adopted.

5.3.5 USB Device (USBD)

This block interfaces the USB Device Controller (namely UDCBLK) via the UDC's application interface and interfaces the FPI Interface block via the SMIF register interface. It consists of the following blocks:

Some blocks need to be redesigned

5.3.5.1 USB Device Interface Block (UDBLK)

The USB Device Interface block consists basically of the USB Device State Machine (UDSM) and the multiplexing logic needed to multiplex the current active transmit/ receive EndPoint channel with the UDSM. The multiplexing is done transparent to the UDSM, i.e the UDSM doesn't even know the existence of the 8 transmit/receive End-Point Channels, it is designed to operate as if there is only one transmit channel from which it reads transmit data and only one receive channel to which it writes data that is received from the UDC. The MUX logic takes care of routing the data to the appropriate transmit or receive channel.

Adaptation to the actual amount of interfaces/endpoints

5.3.5.2 Register Block (RBLK)

The USBD Register Block contains the registers that are defined for the USBD block. This block also contains the RxByteCount[15:0] registers for all the 8 channels. These RxByteCount registers belong logically to the corresponding EndPoint Block of the UDC but they are physially implemented in this block.

Located in the USB clock domain the Register Block has to do the clock rate adaptation for the input signals from the FPI interface block.

Additionally this block contains all the miscellaneous logic that doesn't belong to any other block.

5.3.5.3 FIFO Block (FIFOB)

The FIFO Block is used for clock rate adaptation and data buffering. As the UDC access to memory needs to be granted within 4 clock cycles a local storage is necessary. In addition this block transforms the 8bit UDC application bus to the 32bit SMIF.

Located in the USB clock domain, the FIFO block has to do the clock rate adaptation for the input signals from the FPI interface block.

Logically, the FIFO block contains a transmit FIFO (TXFIFO) or a receive FIFO (RXFIFO) for each EndPoint. Physically, theses FIFOs are implemented by 128x8bit (dualported?) RAM (16 endpoints with 8 byte each) and the corresponding control machine.

New design block

FIFO Control (FIFOC)

The FIFO control block is responsible for maintaining the buffer start and end pointers for each channel independantly. For this it implements one controller for each physical port of the FIFO. As only one endpoint can be active at a given point of time, the controllers load the associated pointers from an internal register array or memory.

The controller on the UDC-side has to guarantee 4-clockcycle access for the UDC. Therefore, UDC transfers have precedence over SMIF transfers. If necessary, pending transfers on the SMIF side have to be stalled, resulting in waitstates on the FPI bus.

New design block

5.4 Operating Modes